In step 903, the dictionary function selects one of a plurality of lookup objects in correspondence to the transmitted user parameter. An example of this step is described in more detail above with reference to Fig. 5b.

In step 904, the dictionary function returns, to the resource program, a lookup object identifier that identifies the detected lookup object corresponding to the user parameter.

In step 905, the resource program transmits a read resource identifier independent of a user parameter to the detected lookup object using the received lookup object identifier.

In step 906, the lookup object retrieves the corresponding resource data.

In step 907, the lookup object transmits the resource data to the client.

In case the resource data correspond to a resource function, steps similar to steps 805, 806 and 807 of Fig. 8 may be performed.

The embodiment of Fig. 9 provides a two-step operation of first identifying an appropriate lookup object corresponding to the user parameter and then retrieving the desired resource data based on the resource identifier from the detected lookup object.

Referring to Fig. 10, that figure illustrates a time sequence of processing steps performed by the resource program in accordance with another embodiment consistent with methods, systems, and articles of manufacture consistent with the present invention. Fig. 10 outlines steps for providing a client with resource data in correspondence to a selected user parameter. Fig. 10 illustrates an embodiment similar to the embodiment described with reference to Fig. 5c, and shows steps involving a client, the resource program and a lookup object. The sequence of processing steps may be executed by the data processing system 600 depicted in Fig. 6, however, Fig. 10 is not limited thereto. The following description of Fig. 10 illustratively refers to the data processing system 600 of Fig. 6.

In step 1001, the user parameter component 181 of the resource program receives a user parameter from the client.

In step 1002, the application component 182 of the resource program reads a resource identifier independent of the user parameter in accordance with the

execution of an application and transmits the user parameter and the resource identifier to the lookup object.

In step 1003, the lookup object retrieves resource data corresponding to the user parameter and the resource identifier.

In step 1004, the lookup object transmits the retrieved resource data to the client.

Referring to Fig. 11, that figure illustrates a time sequence of processing steps a performed by the resource program in accordance with another embodiment consistent with methods, systems, and articles of manufacture consistent with the present invention. Fig. 11 outlines steps for providing a client with resource data in correspondence to a selected user parameter. Fig. 11 shows steps involving a client, the resource program and a lookup object. The sequence of processing steps may be executed by the data processing system 600 depicted in Fig. 6, however, Fig. 11 is not limited thereto. The following description of Fig. 11 illustratively refers to the data processing system 600 of Fig. 6.

In step 1101, the user parameter component 181 of the resource program receives a user parameter from the client.

In step 1102, the resource program receives resource data (e.g., data in a local format or language) from the client. The resource data may, for example, be a floating point number in a local format.

In step 1103, the resource program transmits the user parameter and the resource data to the lookup object.

In step 1104, the lookup object determines a resource identifier (e.g., a resource function) to be applied to the resource data.

In step 1105, the lookup object transmits the resource identifier (or the resource function) to the resource program.

In step 1106, the resource program then converts the received resource data (e.g., a floating point number) in a "local" format into a format internal to the resource means, and independent of the user parameter. The resource program can perform the conversion by using a look up object.

The steps outlined with reference to Fig. 11 provide conversion of a user input dependent on a user parameter, i.e. a "localized" input, into a form which is independent of the user parameter and which may be used by the resource program for further processing.

As described in the above embodiments, for example, when writing web applications or web pages which may be used by users from different countries or localities, the applications may be developed independently of the intended users and may then be "localized" for the different countries or localities.

Thus, in an environment of, for example, Microsoft® Active Server® Pages and the Visual Basic® Scripting language, a set of functions can be used to add "internationalization support". Instead of copying active server pages and then translating these copied pages into different formats or languages, the functions can convert a generic active server page (e.g., a server page independent of user parameters or user environments) into a representation for display at a user which is dependent on a selected user parameter or environment.

The above-described functions provide support for localized text resources, provide for conversion of locally different user inputs like date and time into internal objects and for storage of the data in these objects, and further provide conversion of these objects into a localized format for a user.

Prior to using the functions, the user parameter must be defined. The user parameter can be defined by offering the user a web page with a language selection. This information can be stored in a session object, or it may also passed to following pages in a URL.

When localized text resources are provided, after the user has selected a language by selecting a user parameter, the application loads the text resources required for the application in a lookup object. The lookup object may map a resource description, i.e. the resource identifier, to the actual resource text, i.e. the resource data. The lookup object may be stored in an application object (or session object) and may be identified by a language identifier.

After initializing the lookup object, an active server page application may use the lookup function taking a resource identifier as a parameter and returning a resource data depending on the user parameter. Further, a function may check